Lec 5 on pdf: lec 4

← ال (Finite automata) ، (RE) ← فيهم وجه وظهور.
مبيعملوش (support) لـ (list of instruction).

← ال (lexical) بياخد ال (string of characters) يحولها لـ (stream of tokens) وتكون هي دخل ال (Parsal).

وظائف ال Parsal

١- ياخد (stream of tokens) يحولها لـ (Parse tree).
٢- باقي الوظائف Page ١٠.

stmt
if stmt , else stmt end
While stmt do stmt .... end

Context Free grammers

← بياخدي عشان نعمل (describing the recursive structure)

ـــ الـ(set of terminals) ــــ ده الشكل اللى عليه ~~الكلمات~~ ~~الكلمات~~ بيمثلوا اللغة بتاعتى.

⟶ set of Production

symbol ⟶ list of sym.

ـــ دى (Grammer) كله بـ (support) لـ (Production tools)

وظيفة الـ (Production tools) بتعمل (replace) ~~...~~ فى الـ (string) الناتج.

symbol ⟶ list of sym.

S ⟶ T ∪ NT ∪ ε

(NT ↓ non terminal)

X

Y

A

ـــ هنشيل الـ (NT) ونحط مكانه

ما مكانه عبارة عن ε أو T أو NT.

Ex

$S \longrightarrow Ab$

$A \longrightarrow Bc$

$B \longrightarrow Cc$

$C \longrightarrow \varepsilon$

$T = \{b, c\}$ ← ال (strings) بتاعة اللغة بتاعتي

مبتقبلش غير b's و c's.

$NT = \{S, A, B, C\}$ ← واحده منهم ميزه (انها) (start state)

مش شرط تكون S لكن (start state) هيا أول واحدة.

$S \Rightarrow Ab \Rightarrow Bcb \Rightarrow Cccb \Rightarrow \varepsilon ccb$

terminals ← b's و c's.

← هنجيب ال (Grammer) الناتج ده و نشوف ال (structure)

في البداية ينفع نعمله (driven) لو وصلك لل (Grammer) ده تبقى Accepted.

(id) or id + id or (id + id)

or id * id or (id * id) or id + id + id

← كل ال (expressions) ← (arithmetic expressions)
مقبولة.

← ازاى أعمل (check).

← ال (Grammer) يعرف كل ال (expression) اللى
بتوصف اللغة بتاعتى.

← و ال (Grammer) يعمل (exclude) لكل ال (exp.)
اللى مش معايا.

id ← لو موجوده كده ⇐ ال id تعتبر (terminal).

id → int ⇐ كده ال id تعتبر (non terminal)
لـ واللغة دى هتقبل أرقام بس.

Page 17

← ازاى نعمل (driven) لأى expression

← مقدرش أعمل (replacement) لـ (terminal).
بواسطة ال (Production tool).

← كل أما أدخل لـ (terminal) يكون ثابت معانا
مش بيتشال.

$\{ (^i)^i \mid i \geq 0 \}$ ← عاوز لغة توصف

$S \longrightarrow (S)$ ← يعني كل قوس هيتفتح لازم يتقفل بعده قوس.

$S \longrightarrow \varepsilon$

← ده بيحقق شرط ال(balanced) بس ده مش (nested) ده ماشي (Cascaded).

or $S \longrightarrow (A$

$A \longrightarrow)$

(A(A(A

( ) ( ) ( )

← لو عايز يعمل (drivation) للآخر وسط Page 24

if if id then id else id fi then id else id fi

Expr $\xrightarrow{1}$ if Expr then Expr else Expr fi

$\xrightarrow{1}$ if (if Expr then Expr else Expr fi)

then Expr else Expr fi

* ③ → if if id then id else id fi then id else id fi;

Quiz

x (b , x (a

abcba

S → ...

S → Aa

A → Bb

B → Cc

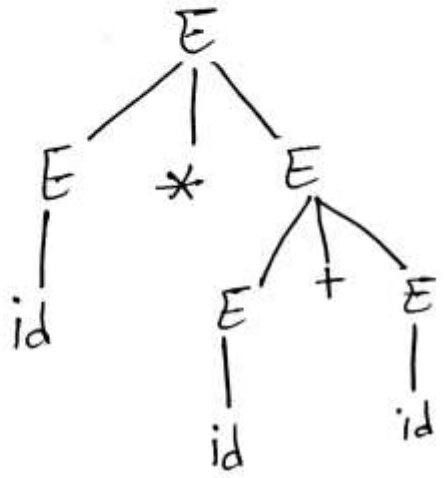
Lex - bison - Yac → إعمل (report) عن أي واحد منهم.

Lex ← (tool) بتعمل (lexical).

ال Parsing

- بيعمل (derivation)
- بيطلع (Parse tree).

~~لما نطبق ال (Product) مرة في كل~~



$E \Rightarrow E * E$

$E * E + E \overset{*}{\Rightarrow}$

id * id + id

← مختلف في ال (derivation) وال (Parsing tree) للرسمة في Page 37 .

← معناها ان انا ممكن ممكن ادخل بأكثر من (derivation)

← بس في الرسمة اللي فوق انتي وصلت لها بشكلين وده مشكلة سمي Ambigous .

← بالنسبة للرسمة اللي فوق

$E \Rightarrow E + E \Rightarrow id * E$

$\Rightarrow id * E \overset{+}{*} E \Rightarrow$ (Leftmost deri.)

$\Rightarrow id * id \overset{+}{*} E \Rightarrow$

ال E نستبدلها ب E + E

id * id + id

Right most ← بشتغل على ال (non terminal) في
ال Right واعمله replacement

← فرقه كويس بين ال right most ، left most.

← في الآخر هتوصل لنفس النتيجة لكنه الاختلاف في الأوامر
اللي في الذهن.

← في ال quiz الحل رقم (c) عالشمال.

S
⋮
abcbdca

Quiz 2 ← الأخيرة عالميين.

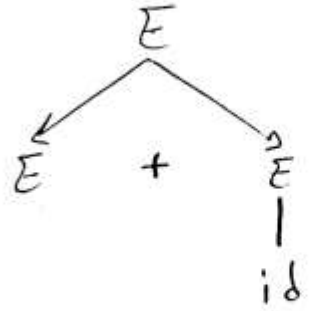
Ambiguous ← لما يكون عندي (2 derivation) في string
يطلعوا أكثر من Parse tree.

← طريقة معالجته ← المرة الجاية.

$S \longrightarrow SS$

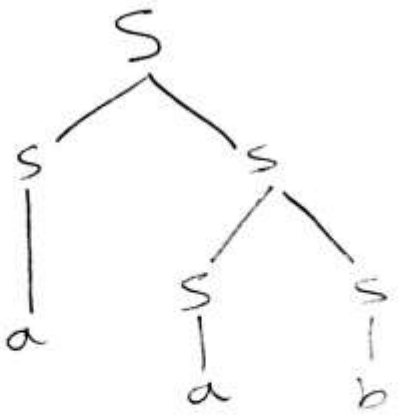
$SS \longrightarrow a$

$a \longrightarrow b$

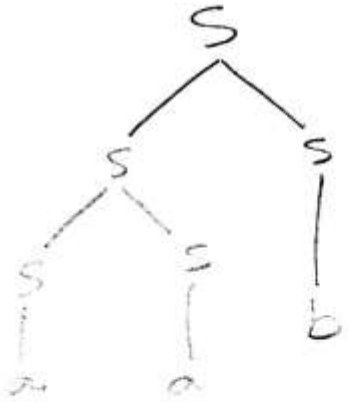


Quiz

a) $S \longrightarrow SS \mid a \mid b$



(right associative)

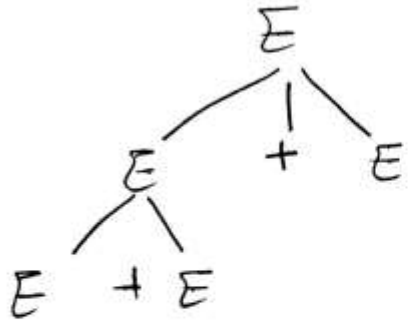


(left Assoc.)

Ambigous ←

b) $E \longrightarrow E + E \mid id$

Ambigous



c) $S \rightarrow Sa \mid Sb$

← استحالة يكون Ambigous.

d) $E \rightarrow E' \mid E' + E$

$E' \rightarrow -E' \mid id \mid (E)$

← هو هنا فصل الحاجة اللي هتخليه Ambigous